

Conversion program outputs BCD values

MIKE MCGLINCHY, MDM DESIGNS, SAN BRUNO, CA

A program entitled HEXBCD is an 8051 μ C subroutine that converts a 16-bit HEX number in internal RAM locations 33 and 34H into its BCD equivalent (Listing 1). The program places the result in scratchpad registers R1, R2, and R3. (You can download the program from EDN's Web site, www.edn-mag.com. At the registered-user area, go into the Software Center to download the file from DI-SIG, #2120.)

Many applications require a μ C to output BCD values,

such as for driving a BCD seven-segment LED decoder. Converting an 8-bit number to BCD is relatively simple; the routine LOW8, which is part of the main HEXBCD program, performs an 8-bit conversion. You can add more registers to expand the overall conversion program beyond 16 bits to handle larger numbers.

The maximum time to perform a 16-bit conversion is 236 instruction cycles. With a 12-MHz crystal and the corre-

LISTING 1—HEXBCD CONVERSION PROGRAM

```

HEXBCD:
    CLR A                ;zero BCD result
    MOV R1,A             ;registers R1, R2 and R3.
    MOV R2,A
    MOV R3,A
    MOV A,33H            ;Get HIGH byte of hex #.
    JNB ACC.7,BIT6       ;If bit 7 of HIGH byte
                        ;is NOT high, test next bit.
                        ;It's HIGH, so put in 32768H.
    MOV R1,#003H
    MOV R2,#027H
    MOV R3,#068H

;-----
BIT6:  JNB ACC.6,BIT5     ;If bit 6 of HIGH byte is
                        ;NOT high, test bit 5.
                        ;It's HIGH, so
    MOV R4,#001H
    MOV R5,#063H
    MOV R6,#084H
    CALL BCDADD           ;Add 16384H to R1, R2 and R3.
;-----
BIT5:  JNB ACC.5,BIT4     ;If bit 5 of HIGH byte is
                        ;NOT high, test bit 4.
                        ;It's HIGH, so add 8192H to
                        ;R1,R2 and R3.
    MOV R5,#081H
    MOV R6,#092H
    CALL BCDADD
;-----
BIT4:  JNB ACC.4,BIT3     ;If bit 4 of HIGH byte is
                        ;NOT high, test bit 3.
                        ;It's HIGH, so add 4096H
                        ;to R1,R2 and R3.
    MOV R5,#040H
    MOV R6,#096H
    CALL BCDADD
;-----
BIT3:  JNB ACC.3,BIT2     ;If bit 3 of HIGH byte is
                        ;NOT high, test bit 2.
                        ;It's HIGH, so add 2048H
                        ;to R1, R2 and R3.
    MOV R5,#020H
    MOV R6,#048H
    CALL BCDADD
;-----
BIT2:  JNB ACC.2,BIT1     ;If bit 2 of HIGH byte is
                        ;NOT high, test bit 1.
                        ;It's HIGH, so add 1024H
                        ;to R1, R2 and R3.
    MOV R5,#010H
    MOV R6,#024H
    CALL BCDADD
;-----
BIT1:  JNB ACC.1,BIT0     ;If bit 1 of HIGH byte is
                        ;NOT high, test bit 0.
                        ;It's HIGH, so add 512H
                        ;to R1, R2 and R3.
    MOV R5,#005H
    MOV R6,#012H
    CALL BCDADD
;-----
BIT0:  JNB ACC.0,LOW8     ;If bit 0 of HIGH byte is
                        ;NOT high, test lower byte.
                        ;It's HIGH, so add 256H
                        ;to R1, R2 and R3.
    MOV R5,#002H
    MOV R6,#056H
    CALL BCDADD

;-----
; LOW8-Does the lower 8 bits hex-bcd conversion
;
LOW8:  MOV A,34H          ;Get low byte of hex # from
                        ;loc 34H.
    MOV B,#100            ;find out how many 100's.
    DIV AB
    MOV R5,A              ;R5 holds BCD hundreds
    MOV A,#10
    XCH A,B               ;find out # of 10's.
    DIV A
    SWAP A
    ADD A,B
    MOV R6,A              ;R6 holds BCD 10's and
                        ;1's digits.
    CALL BCDADD           ;add lower byte BCD # to
                        ;upper byte results
                        ;answer in R1, R2 and R3.
    DONE: RET             ;16 bit conversion complete!

;-----
; BCDADD-adds R4,R5 and R6 to R1, R2 and R3
;-----
BCDADD:
    PUSH ACC              ;Save Acc.
    MOV A,R6              ;Get low BCD digits
    ADDC A,R3
    DA A
    MOV R3,A              ;low digits BCD answer in R3

    MOV A,R5              ;Get middle BCD digits
    ADDC A,R2
    DA A
    MOV R2,A              ;middle digits BCD answer in R2

    MOV A,R4              ;Get high BCD digit
    ADDC A,R1
    DA A
    MOV R1,A              ;high digits BCD answer in R1

    MOV R4,#000H          ;Zero R4 in case it's not
                        ;used again.
    CLR C                 ;Clear Carry flag
    POP ACC               ;restore Acc.
    RET

;-----
END

```

sponding 1 $\mu\text{sec}/\text{instruction}$, the maximum time is then 236 μsec . The more 1s that exist in the number, the longer the conversion takes. For example, converting FFFFH (all 1s) to its BCD equivalent (65,535) takes the maximum of 236 μsec . Converting 4104H to BCD takes 113 μsec because this HEX number contains only three 1s.

The program begins by clearing the BCD registers R1, R2, and R3. The program then takes the high byte of the 16-bit HEX number from RAM location 33H and tests each bit for a high. If a bit is high, the program places a corresponding binary-weighted number in temporary-storage registers R4, R5, and R6. For example, if bit 6 (bit 14 overall) of the high

byte is high, the program places 16384H in R4, R5, and R6. The program then calls a subroutine, BCDADD, which adds 16384H to the contents of R1, R2, and R3 using the ADDC A,R1 (add register to accumulator with carry) and DA A (decimal adjust accumulator) instructions.

After the program tests and adds the upper 8 bits, the intermediate BCD answer is in locations R1, R2, and R3. The routine LOW8 then does a quick HEX-to-BCD conversion of the lower 8 bits. The program adds this result to the high-byte result by calling BCDADD once more. (DI #2120) **EDN**

To Vote For This Design, Circle No. 412

Edge detector runs off of single supply

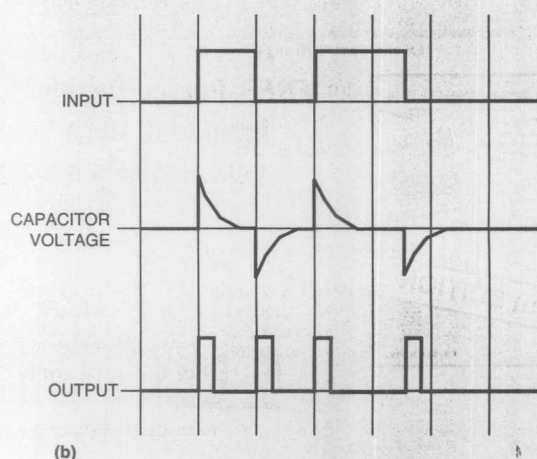
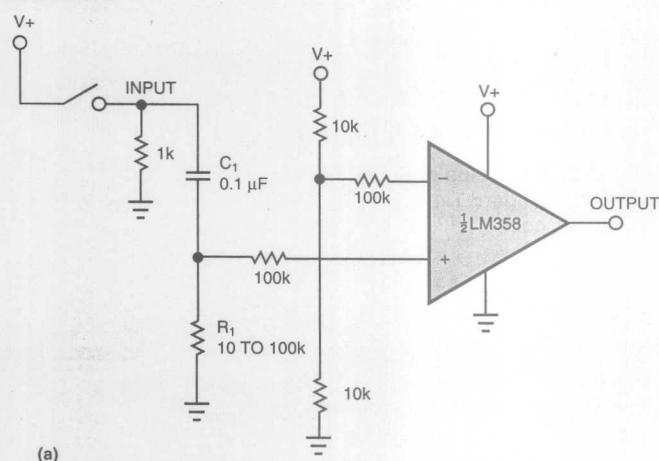
JOE MELOCHE, INGERSOLL-RAND FLUID PRODUCTS DIVISION, BRYAN, OH

You can use a single-supply circuit to generate a pulse on both the rising and the falling edges of a signal for use with counters or similar devices that require only a rising edge to trigger (Figure 1a). R_1 and C_1 form a differentiator that converts rising edges to positive spikes and falling edges to negative spikes. This output drives the LM358, which the circuit configures as a comparator with a reference input equal to $\frac{1}{2}V_+$. As expected, the LM358 converts the positive spike to a positive square pulse of approximately 2.5τ in duration,

where $\tau = R_1 \times C_1$. The LM358 also converts the negative spikes to similar positive pulses because of the LM358's characteristic ability to operate as a fully functional single-supply op amp (Figure 1b). Tests using LM358s from National Semiconductor (Santa Clara, CA) and Motorola (Austin, TX) confirm proper operation. (DI #2116) **EDN**

To Vote For This Design, Circle No. 413

FIGURE 1



A single-supply op amp (a) operating as a comparator teams with differentiator R_1/C_1 to produce positive pulses at both rising and falling edges of the input (b).